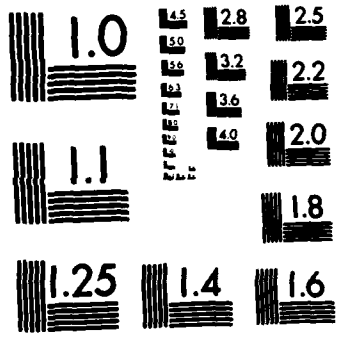
AD-A148 207 PROBABILISTIC ANALYSIS OF ALGORITHMS FOR NP-COMPLETE PROBLEMS(U) CASE WESTERN RESERVE UNIV CLEVELAND OHIO DEPT OF COMPUTER ENGINEERING J FRANCO OCT 84 1/1

UNCLASSIFIED AFOSR-TR-84-1005 AFOSRJ-82-0331 F/G 12/1 NL

END

FILMED

DTIC



AFOSR-TR- 84-1005

(4)

UNITED STATES AIR FORCE
AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
BUILDING 410, BOLLING AFB, D.C. 20332

Grant No. AFOSR 82-0331

Final
[redacted] SCIENTIFIC REPORT

Probabilistic Analysis of Algorithms for NP-complete Problems

John Franco, Principal Investigator
Department of Computer Science
Indiana University
Bloomington, Indiana 47405

Abstract

The probabilistic performance of a number of algorithms for the Satisfiability Problem (SAT) has been investigated analytically and experimentally using a constant-clause-size model generating n clauses of k literals taken from r variables as well as a constant-density model generating n clauses containing each of r variables independently with probability p. In the case of the constant-density model one algorithm has been shown to solve SAT in polynomial time with probability approaching 1 as n and r get large when $p > \ln(n)/r$ and another has been shown to solve SAT in polynomial time with probability approaching 1 as n and r get large when $p < \ln(n)/(2r)$. In the case of the constant-clause-size model the unit clause heuristic has been shown to be effective, in probability, when $\lim_{n,r\to\infty} n/r < 2**(k-1)((k-1)/(k-2))**(k-2)/k$ and another heuristic has been shown to be effective, in probability, when $\lim_{n,r\to\infty} n/r < (1+\ln(k-1))\cdot 2**(k-2)/k$ for $k > 3$. When $k = 3$ the unit clause heuristic with the next variable given an assignment which satisfies the maximum number of clauses has been shown effective, in probability, when $\lim_{n,r\to\infty} n/r < 3$. Experiments have shown the existence of other algorithms which perform better, in the probabilistic sense, than the algorithms analysed.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 84-1005
6a. NAME OF PERFORMING ORGANIZATION Case Western Reserve University 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Air Force Office of Scientific Research
6c. ADDRESS (City, State and ZIP Code) Department of Computer Engineering Cleveland OH 44106	7b. ADDRESS (City, State and ZIP Code) Directorate of Mathematical & Information Sciences, Bolling AFB DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFOSR 8b. OFFICE SYMBOL (If applicable) NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AFOSR-82-0331
8c. ADDRESS (City, State and ZIP Code) Bolling AFB DC 20332-6448	10. SOURCE OF FUNDING NOS PROGRAM ELEMENT NO. 61102F; PROJECT NO. 2304; TASK NO. A2; WORK UNIT NO.

11. TITLE (Include Security Classification)
"PROBABILISTIC ANALYSIS OF ALGORITHMS FOR NP-COMPLETE PROBLEMS

12. PERSONAL AUTHOR(S)
John V. Franco*

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM 1/9/83 TO 31/8/84	OCT 84	9

16. SUPPLEMENTARY NOTATION
*Now with the Computer Science Dept, 101 Lindley Hall, Indiana University, Bloomington IN 47405-4101.

17 COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB GR	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The probabilistic performance of a number of algorithms for the Satisfiability Problem (SAT) has been investigated analytically and experimentally using a constant-clause-size model generating n clauses of k literals taken from r variables as well as a constant-density model generating n clauses containing each of r variables independently with probability p. In the case of the constant-density model one algorithm has been shown to solve SAT in polynomial time with probability approaching 1 as n and r get large when $p > \ln(n)/r$ and another has been shown to solve SAT in polynomial time with probability approaching 1 as n and r get large when $p < \ln(n)/(2r)$. In the case of the constant-clause-size model the unit clause heuristic has been shown to be effective, in probability, when $\lim_{n,r\to\infty} n/r < 2^{**}(k-1)(k-1)/(k-2)^{**}(k-2)/k$ and another heuristic has been shown to be effective, in probability, when $\lim_{n,r\to\infty} n/r < (1+\ln(k-1))\ 2^{**}(k-2)/k$
(CONTINUED)

20. DISTRIBUTION AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS ☐	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
CPT John P. Thomas, Jr.	(202) 767-5026	NM

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

84 11 26 115

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

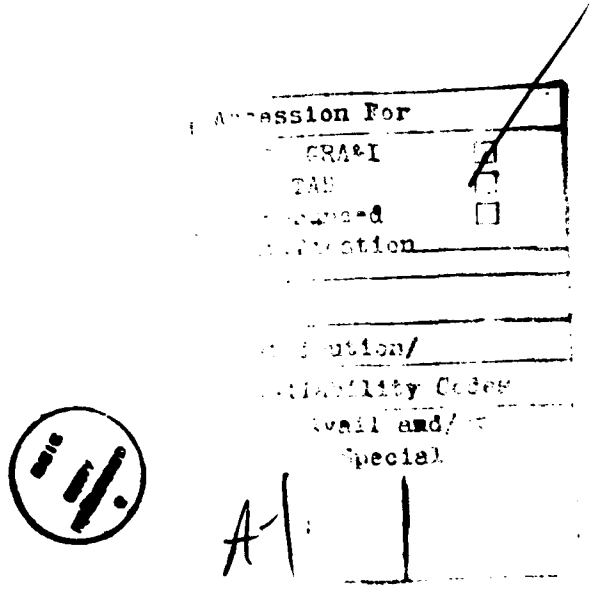
ITEM #19, ABSTRACT, CONTINUED: for k 3. When k=3 the unit clause heuristic with the next variable given an assignment which satisfies the maximum number of clauses has been shown effective, in probability, when $\lim_{n,r\to\infty} n/r < 3$. Experiments have shown the existence of other algorithms which perform better, in the probabilistic sense, than the algorithms analysed.

1. Research Objective

The goal of this research is to develop and analyze algorithms which can, in some practical sense, solve certain NP-complete problems efficiently. By solve we mean determine whether a solution to a given instance of an NP-complete problem exists where, for the problems we have considered, a solution is an assignment of values to a list of variables which cause some predicate to be true. We do not consider actually finding solutions when they exist since doing so adds unnecessary complexity to the statement of the algorithms: the algorithms we consider can all be modified to find solutions without significantly altering performance. NP-complete problems are found in Crytology, Operations Research, Artificial Intelligence, Computer System Design and many other areas. There is no known algorithm for any NP-complete problem which runs in time bounded by a polynomial on the length of the input (polynomial time) in the worst case nor is one likely to be found. We seek algorithms which solve nearly every instance of specific NP-complete problems in polynomial time.

To prove an algorithm A solves nearly every instance of a specific problem X in polynomial time we establish a probability distribution $D(n)$ on instances of X of "size" n (referred to as a model) and then show that A solves a random instance of X generated according to $D(n)$ in polynomial time with probability approaching 1 as n approaches infinity; then A is said to solve X efficiently in probability. Usually the proof holds only under certain conditions. Sometimes, when $D(n)$ is such as to heavily favor the generation of instances with solutions, the weaker result that A "proves" the existence of a solution to a random instance of X in polynomial time with probability bounded from below by a constant greater than zero is obtained instead; then A is said to solve X efficiently with bounded probability. Again, the result holds only under certain conditions (one condition that must be satisfied is that nearly all random instances generated according to $D(n)$ have a solution). The algorithms that we consider here "prove" the existence of a solution by repeatedly choosing a variable and an assignment to that variable until the predicate is true: at each step the possible choices are ranked based on some heuristic and the top ranked possibility is chosen. For the kinds of algorithms and distributions we consider, if A solves X efficiently with bounded probability under some set of conditions then we may regard this as strong evidence that the *Backtrack* algorithm, using the heuristics of A to decide the order in which to consider variables and assign values, solves X efficiently in probability under the same set of conditions.

The NP-complete problem we are primarily interested in is the Satisfiability problem (SAT). An instance *I* of SAT is a boolean expression in conjunctive normal form and a solution to that instance, if one exists, is a truth assignment to the variables in *I* which cause *I* to have value true; such a truth assignment is said to satisfy *I*. SAT remains NP-complete even if all disjunctions contain as few as three literals. SAT is closely related to problems in Artificial Intelligence as well as other NP-complete problems. Algorithms which solve SAT efficiently in some probabilistic sense will, with slight modification, probably solve other NP-complete problems efficiently in the same probabilistic sense.



2. Status of the Research

Although there has been a significant level of research activity in this area no one has succeeded in getting the results we have obtained for algorithms designed to solve instances of SAT efficiently in some probabilistic sense.

Two models have been used for analysis: one is a constant-clause-size model and the other is a constant-density model. According to the constant-clause-size model a random instance of SAT contains n clauses (disjunctions) selected independently and uniformly from the set of all possible disjunctions containing exactly k literals which can be composed from r boolean variables under the restriction that no two literals in the same disjunction are associated with the same variable. We are interested in the case $k \geq 3$ since SAT is NP-complete if clauses are allowed to have three or more literals. For the special case $k = 3$ SAT is called 3-SAT. According to the constant-density model a random instance of SAT contains n clauses each generated independently as follows: for each of r variables (a) place into the clause, with probability $p/2$, the uncomplemented literal associated with the variable, (b) place into the clause, with probability $p/2$, the complemented literal associated with the variable and (c) place neither complemented nor uncomplemented literal into the clause with probability $1 - p$.

The following two algorithms solve SAT efficiently in probability under the constant-density model when n and r are polynomially related. Let I be a random instance of SAT.

A1: Repeat
    Randomly choose a truth assignment t for the variables in I
Until t satisfies I
Output("a solution exists")

A2: Search I for a null clause
If a null clause was found Then Output("no solution exists")
    Else Output("cannot determine whether a solution exists")

In [3] it is shown that A1 solves SAT efficiently in probability when $p > \ln(n)/r$ and A2 correctly determines that no solution exists for a random instance of SAT with probability approaching 1 as n and r approach infinity when $p < \ln(n)/(2r)$. Thus, under the constant-density model, SAT is solved efficiently in probability by algorithms A1 and A2 for all but a vanishingly small range of values of p if n and r are polynomially related (it is easy to see why this is a reasonable restriction). Although these results are theoretically interesting they have little practical meaning since it is unlikely that a random truth assignment satisfies a random instance of SAT and that a random instance of SAT contains a null clause. The results obtained for the constant-clause-size model are probably more meaningful.

Assuming the constant-clause-size model, the algorithms below solve SAT efficiently with bounded probability under various conditions. The algorithms are defined using the following symbols, terms and functions. Let $V = \{v_1, v_2, ..., v_r\}$ be the set of r variables from which clauses are composed and let $L = \{v_1, \bar{v}_1, ..., v_r, \bar{v}_r\}$ be the set of $2r$ literals associated with V (the set of literals contained in a clause c is a subset of L such that for all $1 \leq i \leq r$ both v_i and $\bar{v}_i$ do not appear in c). For every $u \in L$, $var(u)$ is the variable associated with u (for example, $var(v_1) = var(\bar{v}_1) = v_1$). For all $1 \leq i \leq r$, $v_i \in L$ and $\bar{v}_i \in L$ are said to be complementary literals. For every $u \in L$, $comp(u)$ is the literal in L which is complementary to u (for example, $comp(\bar{v}_1) = v_1$). If a clause contains only one literal it is called a unit clause and a unit clause may be regarded as being a literal. Let I be a random instance of SAT generated according to the constant-clause-size model. Let $| u |$ denote the number of occurrences in I of literal u.

A3: Repeat
    If there is a unit clause l in I Then $u \leftarrow l$
        Else choose u randomly from L
    Remove from I all clauses containing u
    Remove from I all occurrences of $comp(u)$
    $L \leftarrow L - \{u, comp(u)\}$
Until I is empty or there exist two complementary unit clauses in I
If I is empty Then Output("a solution exists")
        Else Output("cannot find a solution")

A4: Repeat
    If there is a unit clause l in I Then $u \leftarrow l$
        Else begin
            Choose v randomly from V
            If $|v| > |\bar{v}|$ then $u \leftarrow v$ else $u \leftarrow \bar{v}$
        End
    Remove from I all clauses containing u
    Remove from I all occurrences of $comp(u)$
    $V \leftarrow V - \{var(u)\}$
    $L \leftarrow L - \{u, comp(u)\}$
Until I is empty or there exist two complementary unit clauses in I
If I is empty Then Output("a solution exists")
    Else Output("cannot find a solution")

A5: Repeat
    Let c be a smallest clause in I
    Choose u randomly from c
    Remove from I all clauses containing u
    Remove from I all occurrences of $comp(u)$
Until I is empty or there exist two complementary unit clauses in I
If I is empty Then Output("a solution exists")
    Else Output("cannot find a solution")

In [4] the results below are obtained based on the constant-clause-size model. In this analysis the parameter k is assumed fixed and n and r are allowed to grow toward ∞.

1. There exists a constant d and a constant $e \geq d$ such that a random instance of SAT has a solution with probability approaching 1 as $n, r \rightarrow \infty$ if $\lim_{n,r\rightarrow\infty} n/r < -d/\ln(1-2^{-k})$ and a random instance of SAT has no solution with probability approaching 1 as $n, r \rightarrow \infty$ if $\lim_{n,r\rightarrow\infty} n/r > -e/\ln(1-2^{-k})$.

2. Neither A1 nor A2 solve SAT efficiently in probability or with bounded probability for any fixed (function of k) limiting ratio n/r

3. Algorithm A3 solves SAT efficiently with bounded probability if

$$\lim_{n,r\rightarrow\infty} n/r < \frac{2^{k-1}}{k}\left(\frac{k-1}{k-2}\right)^{k-2} \quad \text{for } k \geq 3$$

and does not solve SAT with probability approaching 1 when

$$\lim_{n,r\to\infty} n/r > \frac{2^{k-1}}{k}\left(\frac{k-1}{k-2}\right)^{k-2} \quad for\ k \geq 3$$

4. Algorithm A4 solves SAT efficiently with bounded probability if

$$\lim_{n,r\to\infty} n/r < 2.8 \quad when\ k = 3$$

and does not solve SAT with probability approaching 1 when

$$\lim_{n,r\to\infty} n/r > 2.8 \quad and\ k = 3$$

5. Algorithm A5 solves SAT efficiently with bounded probability if

$$\lim_{n,r\to\infty} n/r < \frac{2^{k-2}}{k}(1 + \ln(k-1))$$

for the case $k > 3$ and if

$$\lim_{n,r\to\infty} n/r < 3 \quad for\ the\ case\ k = 3$$

The two algorithms below have been studied experimentally using the constant-clause-size model with $k = 3$. These algorithms dynamically assign weights to each literal in L and these weights are used to select the next literal. The weighting functions are defined as follows:

$$\omega(l) = \sum_{c\in S(l)} 2^{-size(c)}$$

$$\omega_E(l) = \prod_{c\in P(l)} (1 - 2^{-size(c)})$$

where $S(l)$ is the collection of clauses in I containing literal l, $P(l)$ is the collection of clauses resulting from removing clauses containing l and all occurrences of $comp(l)$ from I and $size(c)$ is the number of literals contained in clause c if c has not been removed from I and $size(c) = \infty$ otherwise. It should be noted that $size(c) = k$ for all c in a random instance of SAT generated according to the constant-clause-size model but $size(c)$ changes as literals are removed or when c contains the next chosen literal in the algorithms below. It should also be noted that $\omega_E(l)$ is a measure of the expected number of solutions that exist for an instance of SAT with $var(l)$ set so that literal l has value true; choosing a truth assignment to maximise the expected number of solutions to the remainder of the instance seems to be a reasonable heuristic and is the basis for algorithm A7.

A6: Repeat
$u \leftarrow l : l \in L,\ \forall l' \in L\ \omega(l) \geq \omega(l')$
Remove from I all clauses containing u
Remove from I all occurrences of $comp(u)$
$L \leftarrow L - \{u, comp(u)\}$
Until I is empty or there exist two complementary unit clauses in I
If I is empty Then Output("a solution exists")
Else Output("cannot find a solution")

A7: Repeat
$u \leftarrow l : l \in L,\ \forall l' \in L\ \omega_g(l) \geq \omega_g(l')$
Remove from I all clauses containing u
Remove from I all occurrences of $comp(u)$
$L \leftarrow L - \{u, comp(u)\}$
Until I is empty or there exist two complementary unit clauses in I
If I is empty Then Output("a solution exists")
Else Output("cannot find a solution")

In [5] the following results of experiments on **A6** and **A7** using the constant-clause-size model to generate random instances are reported:

6. Algorithm **A6** solves SAT efficiently with bounded probability if

$$\lim_{n,r\to\infty} n/r < 3.7 \quad when\ k = 3$$

and does not solve SAT with probability approaching 1 if

$$\lim_{n,r\to\infty} n/r > 3.8 \quad when\ k = 3$$

7. Algorithm **A7** solves SAT efficiently with bounded probability if

$$\lim_{n,r\to\infty} n/r < 3.6 \quad when\ k = 3$$

and does not solve SAT with probability approaching 1 if

$$\lim_{n,r\to\infty} n/r > 3.7 \quad when\ k = 3$$

8. A random instance of SAT generated according to the constant-clause-size model has no solution with probability approaching 1 when $n, r \to \infty$ if

$$\lim_{n,r\to\infty} n/r < 4 \quad \textit{when } k = 3$$

9. A random instance of SAT generated according to the constant-clause-size model has a solution with probability approaching 1 when $n, r \to \infty$ if

$$\lim_{n,r\to\infty} n/r > 4 \quad \textit{when } k = 3$$

3. Interpretation of Results

The constant-clause-size model seems to generate non-trivial instances of SAT since the simple-minded algorithms **A1** and **A2** which work so well on instances generated by the constant-density model do not work at all well on random instances generated according to the constant-clause-size model when the limiting ratio of n/r is fixed (i.e. a function of k). The case of the limiting ratio of n/r being fixed is important since random instances are "hardest" when the probability that a solution exists is about 1/2 and this occurs when the limiting ratio is fixed. Despite the relatively "hard" instances generated by the constant-clause-size model a number of algorithms have been shown to "prove" that a solution to a given random instance I of SAT exists for nearly every I that has a solution when $k = 3$; these algorithms are not quite as effective for arbitrary k.

Perhaps surprising is the difference in the range of n/r over which algorithms perform well probabilistically. In particular, **A3** and **A5** are not much different in structure but the bound on the limiting ratios of n/r for which good probabilistic performance is achieved is larger for **A5** by a factor of about $\ln(k)$. Furthermore, from a previous result [2], the bound on ratios n/r for which good probabilistic performance of the pure literal heuristic is achieved does not even increase with k while the bounds for **A3**,**A4** and **A5** are all exponential in k.

We have been able to rank a number of algorithms for solving SAT by their probabilistic performance. One of these algorithms has been shown experimentally to be extemely effective on instances of 3-SAT when those instances have solutions. We have not yet succeeded in producing an algorithm for SAT which, under the constant-clause-size model, is effective in determining that no solution exists when its input is an instance with no solution. This is the next step in this research. After this we intend to apply the algorithms and analysis mentioned here to other NP-complete problems.

4. Publications

[1] J. Franco and M. Paull, "Probabilistic analysis of the Davis-Putnam procedure for solving the Satisfiability problem," *Discrete Applied Mathematics* 5 (1983), pp.77-87.

[2] J. Franco, "Probabilistic analysis of the pure literal heuristic for solving the Satisfiability problem," to appear in *Annals of Operations Research* (1984).

[3] J. Franco, "Sensitivity of probabilistic results for algorithms for NP-complete problems to input distribution," submitted to *SIGACT News*.

[4] M.T. Chao and J. Franco, "Practical algorithms for the Satisfiability problem," manuscript in preparation.

[5] M.T. Chao, "Probabilistic performance of algorithms for the Satisfiability problem," Ph.D. thesis, Case Western Reserve University, Cleveland, Ohio (1984).

5. Talks

[6] J. Franco, D. Solow and H. Emmons, "Duality, Finite Improvement and efficiently solved optimisation problems," at the Midwest Theory Conference, Chicago, Illinois, (1983).

[7] J. franco, "Sensitivity of probabilistic results for algorithms for NP-complete problems to input distribution," Institute for Defense Analysis (1983).

[8] J. Franco, "Probabilistic analysis of algorithms for the Satisfiability problem," Institute for Defense Analysis (1983).

END

FILMED

1-85

DTIC